Report No. NAWCADWAR-95002-4.5

AN IMPROVED RNS DIVISION ALGORITHM

Peter R. Turner, Ph.D.
Mathematics Department
U.S. NAVAL ACADEMY
Annapolis, MD 21402

7 NOVEMBER 1994

FINAL REPORT

Approved for Public Release; Distribution is Unlimited.

Prepared for
Avionics Department
Engineering Division (Code 4.5.5.1)
NAVAL AIR WARFARE CENTER
AIRCRAFT DIVISION WARMINSTER
P.O. Box 5152
Warminster, PA 18974-0591

NOTICES

REPORT NUMBERING SYSTEM - The numbering of technical project reports issued by the Naval Air Warfare Center, Aircraft Division, Warminster is arranged for specific identification purposes. Each number consists of the Center acronym, the calendar year in which the number was assigned, the sequence number of the report within the specific calendar year, and the official 2-digit correspondence code of the Functional Department responsible for the report. For example: Report No. NAWCADWAR-95010-4.6 indicates the tenth Center report for the year 1995 and prepared by the Crew Systems Engineering Department. The numerical codes are as follows.

Code	Department
4.1	Systems Engineering Department
4.2	Cost Analysis Department
4.3	Air Vehicle Department
4.4	Propulsion and Power Department
4.5	Avionics Department
4.6	Crew Systems Engineering Department
4.10	Conc. Analy., Eval. and Plan (CAEP) Department

PRODUCT ENDORSEMENT - The discussion or instructions concerning commercial products herein do not constitute an endorsement by the Government nor do they convey or imply the license or right to use such products.

Reviewed By: Barry Kirsch **Date:** 16 FEB 1995
Author/COTR

Reviewed By: **Date:** 2/21/95
LEVEL III Manager

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
7 NOV 1994

3. REPORT TYPE AND DATES COVERED
FINAL

4. TITLE AND SUBTITLE

AN IMPROVED RNS DIVISION ALGORITHM

5. FUNDING NUMBERS

6. AUTHOR(S)

PETER R. TURNER, PH.D.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Mathematics Department
U.S. NAVAL ACADEMY
Annapolis, MD 21402

8. PERFORMING ORGANIZATION REPORT NUMBER

NAWCADWAR-95002-4.5

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Avionics Department; Engineering Division Code 4.5.5.1)
NAVAL AIR WARFARE CENTER; AIRCRAFT DIVISION WARMINSTER
P.O. Box 5152
Warminster, PA 18974-0591

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

NAWCADWAR P.O.C. — BARRY J. KIRSCH (CODE 4.5.5.1)

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This paper presents a division algorithm for the Residue Number System which is a modification, and improvement on, the recent algorithm of Hitz and Kaltofen (1994). The relative cost of the divisions is substantially reduced rendering the RNS division feasible for computations which are not division-intensive such as the solution of a system of linear equations of small dimension. The advantages of this algorithm over the original work of Hitz and Kaltofen lies simply in using a ceiling function in place of the floor function. This leads to a better and simpler convergence criterion and test, and, more importantly, to a simple scheme for accelerating the potentially slow early iteration of the Newton-iteration-based algorithm.

14. SUBJECT TERMS

RESIDUE NUMBER SYSTEM (RNS)

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. SECURITY CLASSIFICATION OF ABSTRACT
UNCLASSIFIED

20. LIMITATION OF ABSTRACT
SAR

NAWCADWAR-95002-4.5

AN IMPROVED RNS DIVISION ALGORITHM

PETER R TURNER
MATHEMATICS DEPARTMENT, US NAVAL ACADEMY, ANNAPOLIS, MD 21402

Abstract. This paper presents a division algorithm for the Residue Number System which is a modification, and improvement on, the recent algorithm of Hitz and Kaltofen (1994). The relative cost of the divisions is substantially reduced rendering the RNS division feasible for computations which are not division-intensive such as the solution of a system of linear equations of small dimension. The advantages of this algorithm over the original work of Hitz and Kaltofen lies simply in using a ceiling function in place of the floor function. This leads to a better and simpler convergence criterion and test, and, more importantly, to a simple scheme for accelerating the potentially slow early iterations of the Newton-iteration-based algorithm.

1. Introduction. One of the main reasons that the Residue Number System, RNS, has not become the basis of a widely-used general purpose integer or floating-point arithmetic has been the difficulty in performing division and other nonstandard RNS operations. Over recent years much effort has been expended in the attempt to produce a good RNS division algorithm. These fall into several different basic classes. Most of the earlier algorithms used some form of binary conversion, see [10], [2] and [3] for example. The L-CRT [5] was an attempt to circumvent the need for full divisions by using a modified version of the Chinese Remainder Theorem to make efficient scaling a feasible alternative. The division algorithm of [9] is based on a binary search using RNS comparisons. Gamberger [4] was one of the first to use an extended RNS basis to find common factors and scale the operands. The idea of an extended RNS basis was also exploited in the algorithm of Hitz and Kaltofen [6]. Their approach was to simulate a double-length accumulator to perform accurate integer division using a modified Newton iteration to obtain a "reciprocal" of the divisor relative to the range of the base RNS being used. It is this algorithm which provides the basis for the present work.

One of the chief potential drawbacks of their algorithm is that the desire to have a uniform starting point for the reciprocation phase necessarily implies very slow initial convergence to the reciprocal. For an RNS system with a small dynamic range M this $O(\log M)$ first stage is unlikely to be excessively expensive. However for problems where this dynamic range is large this first phase can be sufficiently expensive as to render the system inappropriate for computations where the number of divisions is large.

One area in which RNS arithmetic has proved highly beneficial is in various aspects of signal processing. In [7], [8], [11] and [12] the use of RNS arithmetic for adaptive beamforming problems using Gaussian elimination was investigated. A divisionless form of the algorithm was found to be feasible for small scale problems [7], [8]. However the dynamic range requirements grow too fast to allow this approach to be practical for larger problems. To understand the importance of speeding up RNS divisions we further observe that the number of divisions needed for the conventional Gauss elimination algorithm is $O(N^2)$ (see [1] or any standard Numerical Analysis text). For integer arithmetic without any temporary use of fractional representations the number of divisions rises to $O(N^3)$. Even with divisions included in the algorithm, the wordlengths and time-penalty associated with the divisions limit its usefulness [11], [12]. An efficient RNS division algorithm therefore has great potential in this and other applications.

The algorithm presented here has two major advantages over the original work of [6]. First by using the ceiling function in place of the floor function in the reciprocation phase of the algorithm, a final comparison (requiring a mixed-radix conversion to the extended RNS basis) is avoided. Second, using this modified algorithm has the additional benefit of making it easy to recognize when an iterate is distant from the required reciprocal and therefore to accelerate the convergence. In [6] the need for this acceleration is recognized and is handled by a sequence of comparisons against powers of 2 to find a better starting point for the Newton iteration. This is implemented parallel comparisons and summing the results. Again this will be adequate if the dynamic range is small enough that the necessary hardware is available. That is log M comparisons must be performed simultaneously - each requiring a full base extension via a mixed radix conversion. For all but the smallest of dynamic ranges this is likely to be unrealistic. This is the motivation for accelerating the basic algorithm. The speed-up achieved is by an asymptotic factor of 2 in the slowest part of the algorithm. Numerical results are presented to verify the reduced iteration count of the new algorithm over a moderate range. The expected savings would grow with the dynamic range. The new algorithm does not improve the *order* of the time-complexity but has the combined merits of simplifying the original algorithm and speeding it up most when it is slowest.

In the next section we begin with a brief summary of the notation used here and of the Hitz and Kaltofen algorithm without addressing the specifics of its implementation in RNS arithmetic which is ably detailed in the original paper [6]. In section 3, the convergence analysis of Newton's method for reciprocation is revisited with reference to the particular context of integer arithmetic. The benefits of using the ceiling function in terms of this convergence analysis and for implementation in terms of avoiding magnitude comparison testing are described. Section 4 is devoted to accelerating the early iterations by recognizing the situation where the iterates are well-removed from the desired reciprocal. Numerical experiments to show the savings made by this algorithm are summarized in Section 5. The details of the RNS implementation of the revised algorithm are essentially unchanged and are only addressed explicitly when necessary for clarification of the presentation.

The paper concludes with a summary of the findings together with some discussion of area-complexity and the potential benefits of having several RNS processors operating in parallel.

2. The division algorithm of Hitz and Kaltofen. In this section we study the division algorithm of Hitz and Kaltofen [6] which can be used to obtain both the integer division and the remainder resulting from division of two integers each represented in an extended RNS which plays the role similar to that of a double length accumulator in regular binary arithmetic. No CRT conversions or their equivalent are required by this algorithm. It does however use several RNS base-extension operations. These however can be performed entirely within the RNS processors - provided enough such processors are available. The section begins with a brief summary of our notation and of the Hitz and Kaltofen algorithm which has at its heart a pseudo-reciprocation step.

The basic idea here is that a double length RNS representation is used in the following manner. Let p_1, p_2, ..., p_L, p_{L+1}, ..., p_{2L} be a set of prime numbers. (They only need to be relatively prime for this purpose but we shall need Gaussian primes for our algorithm in order to perform complex RNS arithmetic efficiently.) Assume they are ordered so that $p_i < p_{L+i}$ for each $i = 1, 2, ..., L$. In [6] slightly stronger order assumptions are made but the additional assumptions play no role; indeed even this assumption can be weakened as we will see in discussing the use of this algorithm later.

Let

$$M = \prod_{i=1}^{L} p_i \,, \qquad \overline{M} = \prod_{i=1}^{L} p_{L+i}$$

so that M represents the dynamic range of the *base RNS* and $M\overline{M}$ represents the range of the *extended RNS.* Some important (though immediately apparent) observations are made in [6]:

M, $\overline{M}$ are relatively prime and $M < \overline{M}$.

It follows that $M^{-1} \bmod \overline{M}$ exists. Also the first N moduli in the extended representation of an extended RNS number represent the remainder of that number mod $\overline{M}$

.

It also follows that multiplication of two numbers represented in the base RNS (that is, two numbers smaller than M) cannot overflow the extended RNS range.

The division algorithm then consists of two stages in order to obtain the integer quotient $\lfloor X/Y \rfloor$ and remainder $X \bmod Y$ where X, Y are positive integers in the base RNS range. In the first, the "reciprocal" $\lfloor M/Y \rfloor$ of Y is obtained using a modification of Newton's method. This is then used to generated the desired quotient and remainder.

The reciprocation algorithm is described in [6] as follows.

```
Algorithm RECIP
Input:    $Y$
Output:   $\lfloor M/Y \rfloor$
begin
  $Z_1 \leftarrow 0$
  $Z_2 \leftarrow 2$
  while $Z_1 \neq Z_2$ do
    $Z_1 \leftarrow Z_2$
    $Z_2 \leftarrow \lfloor Z_1 * (2M - Y * Z_1) / M \rfloor$
  if $M - Y * Z_2 < Y$
    then return $Z_2$ else return $Z_2 + 1$
end.
```

The need for, and correctness of, the correction step is established in [6]. The iteration in the loop is just Newton's method in an "integerized" form. The division by M, equality testing and comparison are all achievable using the extended RNS basis in ways which are also detailed in [6]. We shall discuss later only those details which are needed. Once this M-reciprocation has been completed the rest of the division is straightforward. This algorithm is also given in [6]:

```
Algorithm DIVREM
Input:    $X, Y$
Output:   $\lfloor X/Y \rfloor$ and $X \bmod Y$
begin
```

```
    $Q \leftarrow \lfloor X * \mathrm{RECIP}\,(Y)\,/M \rfloor$
    $R \leftarrow X - Q * Y$
    if $R < Y$ then return $Q$, $R$
      else return $Q + 1$, $R - Y$
end.
```

The comparison and division by M are performed as in the RECIP algorithm and again the reader is referred to [6] for the details of their RNS implementations. One aspect of the implementation is worth commenting on however.

The comparison and division by M both require base extensions between the base RNS and the extended RNS and between the *extension* RNS based on p_{L+1}, ..., p_{2L} and the full extended RNS system. These operations require mixed conversion to the corresponding mixed radix systems, MRS. This step is expensive either in time or, as discussed in [6], in hardware in order to reduce the time-penalty. The hardware configuration suggested is an array of $L(L+1)/2$ modular multipliers with L separate binary trees of modular adders and a final MRS carry-lookahead adder. With relatively small RNS bases this degree of parallelism in the processors may be realizable. However the acceleration phase in [6] also requires $\log M$ of these comparison units - each needing the base extension architecture just described.

For the adaptive beamforming solution referred to above, L is typically around 16 and $\log M$ around 120 even for moderate-size problems. The overall requirement would therefore be for some 15,000 modular multipliers. This level of parallelism can probably be better used for purposes other than just speeding up division. In the case of Gaussian elimination, there are often several divisions using the same divisor and so a conventional MRS conversion could perhaps be efficiently pipelined for this purpose. Avoiding the magnitude comparisons that are central to the Hitz and Kaltofen algorithm would also be a significant benefit. In the remaining sections of this paper we establish an alternative form of the algorithm which avoids these comparisons and achieves a lesser speed-up than the $\log M$ comparisons but much more cheaply. Indeed it uses no additional hardware to that of the base algorithm.

3. Improved reciprocation using the ceiling. It is apparent that the critical part of this division algorithm is RECIP and that any savings which can be achieved there are worthwhile. In this subsection we consider two sources of improvement. The first is the elimination of the correction step by using the ceiling rather than the floor function in the iteration. The convergence analysis must be appropriately modified - or simplified - to take account of this. This removes the need for the final comparison which is implemented in extended RNS by using two base extension operations.

In [6], the number of steps of the Newton iteration is also discussed. For convenience of the analysis this is separated into two parts which can be thought of as a first stage of order $O\,(\log M)$ which achieves an iterate with a relative error of less than 75% and a second $O\,(\log\log M)$ which is the usual quadratic convergence behavior of Newton's method. They go on to discuss a hardware-intensive acceleration of the first stage as described above. In our setting this acceleration is unlikely to be practicable and so the first stage constitutes a potentially prohibitive number of iterations to achieve a suitably good estimate that the quadratic convergence takes over.

We begin with a brief review of the convergence theory of Newton's method for the modular reciprocation operation. The most important aspects are that the (real) Newton iteration generates a *strictly increasing* sequence of iterates - after a possible decrease on the first iteration if the initial estimate is too large. In our setting this will imply a strictly increasing *integer* sequence until it converges. For full details of the analysis of Newton's method see a standard Numerical Analysis text such as [1].

The standard (real arithmetic) Newton iteration for finding M/Y where $M > Y$ is just the application of Newton's method to the solution of the equation

$$f(x) = \frac{1}{x} - \frac{Y}{M} = 0$$

With an appropriately chosen starting point this yields the familiar iteration

$$x_{n+1} = x_n \left(2 - \frac{Yx_n}{M}\right) \tag{1}$$

If $x_n < M/Y$ then it is easy to see that $x_{n+1} > x_n$. Also from the conventional error analysis of this iteration, we have

$$x_{n+1} - \frac{M}{Y} = -\frac{Y}{M}\left(x_n - \frac{M}{Y}\right)^2$$

so that $x_{n+1} < M/Y$. It follows that, at least for $n \geq 1$, the sequence (x_n) is strictly increasing. It will converge, eventually quadratically, provided only that $x_1 > 0$ which will be ensured if $x_0 < 2M/Y$. Usually somewhat more restrictive bounds resulting from a first order Taylor expansion are put on the iterates, namely that $x_0 \in \left(\frac{M}{2Y}, \frac{3M}{2Y}\right)$. These bounds also have the advantage of preventing cycles in the ”integerized” iteration we shall be considering.

The Hitz and Kaltofen algorithm [6] generates an integer sequence by simply taking the floor of each iterate. This will also generate an increasing sequence which ”converges” to either $\lfloor M/Y \rfloor$ or $\lfloor M/Y \rfloor - 1$. In this context ”converges” means that successive iterates are *equal.* The simple modification suggested here is that we replace this floor with the corresponding ceiling so that the iterates are given by

$$Z_{n+1} = \left\lceil Z_n \left(2 - \frac{YZ_n}{M}\right)\right\rceil = 2Z_n - \left\lfloor \frac{YZ_n^2}{M} \right\rfloor \tag{2}$$

which can be implemented easily irrespective of which of floor and ceiling is regarded as a primitive function.

Clearly at each stage this iterate will be greater than its counterpart in the original (or indeed in the continuous) algorithm and is again a strictly increasing sequence until ”convergence ” is achieved. The acceleration due to the use of the ceiling here is likely to be slight and should not be thought of as a particularly significant aspect of this modified algorithm.

As with any discrete or integerized algorithm, ”convergence” has a special interpretation in this context. The strictly increasing nature of the sequence guarantees that eventually an iterate is obtained such that

$$Z_n = \left\lceil \frac{M}{Y} \right\rceil$$

at which point either $Z_{n+1} = \lceil \frac{M}{Y} \rceil$ or $Z_{n+1} = \lceil \frac{M}{Y} \rceil - 1 = \lfloor \frac{M}{Y} \rfloor$. That these are the only possibilities follows from the facts that the error is being reduced at each iteration and $x_{n+1} < M/Y$.

Thus eventually the iterates either remain static at $\lceil\frac{M}{Y}\rceil$ or oscillate between $\lceil\frac{M}{Y}\rceil$ and $\lfloor\frac{M}{Y}\rfloor$. Also two successive iterates cannot be equal unless convergence to $\lceil\frac{M}{Y}\rceil$ has been achieved. Similarly, the oscillation occurs if, and only if, $Z_{n+1} = Z_n - 1$. Thus we have a very simple pair of convergence criteria to test. The iteration is halted if

$$\text{either } Z_{n+1} = Z_n \text{ or } Z_{n+1} = Z_n - 1.$$

Thus we have established

THEOREM 1. *The Modified RECIP algorithm below returns the correct value of* $\lceil\frac{M}{Y}\rceil$

```
  Algorithm Modified RECIP
Input:    $Y$
Output:   $\lceil M/Y \rceil$
begin
  $Z_1 \leftarrow 0$
  $Z_2 \leftarrow 2$
  while $Z_1 \neq Z_2$ and $Z_1 - 1 \neq Z_2$ do
    $Z_1 \leftarrow Z_2$
    $Z_2 \leftarrow \lceil Z_1 * (2M - Y * Z_1)/M \rceil$
  return $Z_1$
end.
```

The convergence test and magnitude comparison required by the original RECIP algorithm are thus replaced by a pair of equality tests. (We observe that, of course, the addition or subtraction of 1 in any RNS system is just the corresponding addition or subtraction of 1 in each residue.) An analysis of the frequency with which the oscillation can occur shows that if we regard Y as uniformly distributed in the interval $(0, M]$ then approximately 75% of the time the iteration will converge to $\lceil\frac{M}{Y}\rceil$ without any oscillation.

Note that the modified algorithm above is *not* the final revised algorithm that is the main objective of this paper. However it does represent the basis of the use of the ceiling function in the integerized Newton iteration which is at the heart of the improvements.

We conclude this section with the observation that the algorithm DIVREM in Section 2 also finishes with a magnitude comparison on which is based a possible correction step. In both the original algorithm and the version being developed here, this is avoidable if only the integer part of the result is required and an absolute less than 1 is acceptable. In the original algorithm this is achieved by simply using $\lceil.\rceil$ in place of $\lfloor.\rfloor$ in DIVREM and eliminating the test. The corresponding stage of the revised algorithm would use RECIP(Y) $= \lceil\frac{M}{Y}\rceil$ and then to set $Q \leftarrow \lfloor X * \text{RECIP}(Y)/M \rfloor$. That this yields an absolute error less than 1 follows since

$$\frac{X}{Y} + 1 > \frac{X}{Y} + \frac{X}{M} > \frac{X * \lceil M/Y \rceil}{M} \geq \frac{X}{Y}$$

so that

$$\left\lfloor \frac{X}{Y} + 1 \right\rfloor > \left\lfloor \frac{X * \lceil M/Y \rceil}{M} \right\rfloor \geq \left\lfloor \frac{X}{Y} \right\rfloor$$

and, except when X/Y is an exact integer, $\left\lfloor \frac{X}{Y} + 1 \right\rfloor = \left\lfloor \frac{X}{Y} \right\rfloor + 1 = \left\lceil \frac{X}{Y} \right\rceil$. (If the true result is an integer, then this quotient will be obtained exactly.) If a true, positive, remainder is also required such as in [6], then some correction step similar to that in DIVREM is necessary.

4. Accelerating the early iterations.

A second consequence of the use of the ceiling function in the iteration is that it becomes easy to recognize when the iterates are well-removed from the desired reciprocal - and to make appropriate adjustments for this. The effect is that the number of iterations used in this first phase is reduced by an asymptotic factor of 2.

The acceleration is based on the fact that if $x_n \ll M/Y$ in (1) then $x_{n+1} \simeq 2x_n$ so that the Newton iterates approximately double with each early iteration from a starting value which is too small. It also follows from (1) that $x_{n+1} < 2x_n$ and therefore if the floor function is used as in the original [6] algorithm, we can never have $Z_{n+1} = 2Z_n$. Using the ceiling function allows the possibility that $Z_{n+1} = 2Z_n$. This condition is also an easy one to check since multiplication by 2 is a simple modular operation. Let us consider when this "iterate-doubling" can occur.

From (2) we see that $Z_{n+1} = 2Z_n$ if

$$\left\lfloor \frac{YZ_n^2}{M} \right\rfloor = 0$$

which is to say that

$$Z_n^2 < \frac{M}{Y}$$

or $Z_n < \sqrt{M/Y}$. This suggests that we simply test whether $Z_{n+1} = 2Z_n$ and if this returns "True" then replace Z_{n+1} with $Z_{n+1} = Z_n^2$ - or perhaps some other well-chosen alternative value. We also observe that the quantity Z_n^2 has already been computed and so no additional computation is required beyond the equality test.

This will clearly accelerate the early iterations especially in the situation where $M/Y \gg 1$. If, however, M/Y is very large then several iterations may still be required before iterates larger than $\sqrt{M/Y}$ are reached. There may be significant gains to be made by including a further scaling into the revised Z_{n+1} when doubling has taken place. The rest of this section is devoted to a brief analysis of this.

Temporarily write $c = M/Y$. Our underlying Newton iteration is then

$$x_{n+1} = x_n \left(2 - \frac{x_n}{c}\right)$$

which is the normal iteration for computing the reciprocal of $1/c$.

The modified algorithm RECIP of Section 3 generates iterates 2, 4, 8, ... , $Z_n = 2^n$ until $Z_n > \sqrt{c}$ whereas the further modification outlined above would generate the iterates 2, 4, 16, ..., $Z_n = 2^{2^{n-1}}$, again until $Z_n > \sqrt{c}$. The first of these sequences requires $\left\lceil \frac{1}{2} \log c \right\rceil$ iterations to satisfy this condition after which iterate doubling will not happen and eventually the quadratic convergence will take over. The second sequence takes $\lceil \log(\log c) \rceil$ iterations to reach this point. Thus the slowest part of the algorithm is accelerated from $O(\log c)$ to $O(\log(\log c))$ iterations at the cost of an additional modular operation and an equality test per iteration. This however does not bring the overall operational complexity of the algorithm down to $O(\log(\log c))$since the number of iterations required to attain a value within a fixed relative error will still be $O(\log c)$just as in [6]. What is achieved is an improvement by a factor of about 2 in this part of the algorithm.

The fixed relative precision used in [6] was 3/4. That is, they establish that $O(\log c)$iterations are required to generate an iterate $Z_n > c/4$ after which $O(\log(\log c))$further iterations are required to achieve convergence. Clearly it is desirable to accomplish the first stage with as few iterations as possible.

In the case of iterate-doubling, it is clearly possible to set $Z_{n+1} = \alpha Z_n^2$ where α is a constant to be chosen. The idea is to get beyond the threshold of $\sqrt{c}$ more rapidly without generating an iterate Z_{n+1} which is so large as to set up oscillations by in turn generating very small iterates. How should we choose α?

The elementary function iteration analysis ([1], Section 3.2, for example) of Newton's method for reciprocals implies that quadratic convergence in the present situation is guaranteed for any $Z_0 \in (c/2, 3c/2)$. This in turn suggests that choosing $\alpha = 3/2$ is feasible. This choice will certainly accelerate the initial growth of the iterates when $Z_0 \ll c$. It turns out to be the optimal constant choice for our integer arithmetic setting.

First, if $Z_n < 3c/2$ then it follows that $Z_{n+1} > 3c/4$ and the subsequent relative errors indeed decrease quadratically. To see this consider again the continuous form of the algorithm and write

$$x_n = c \cdot a_n$$

Then the recurrence for the sequence (a_n) is given by

$$a_{n+1} = \frac{x_{n+1}}{c} = \frac{x_n}{c}\left(2 - \frac{x_n}{c}\right) = a_n(2 - a_n)$$

and if $a_n = 1 - 2^{-k}$ then

$$a_{n+1} = \left(1 - 2^{-k}\right)\left(2 - \left[1 - 2^{-k}\right]\right) = 1 - 2^{-2k}$$

Next, a larger value of α could be chosen to accelerate this early growth even further provided only that $Z_{n+1} > c/2$ whenever $Z_{n-1} < \sqrt{c}$ and $Z_n > c$. Therefore α must be such that $x_n = \alpha c$, or $a_n = \alpha$, yields $a_{n+1} > 1/2$. This implies that $\alpha^2 - 2\alpha + 1/2 < 0$ which yields $\alpha \in \left[1 - \sqrt{1/2}, 1 + \sqrt{1/2}\right]$. Since we are interested in $\alpha > 1$, this suggests that the largest acceptable value is $\alpha = 1 + \sqrt{1/2} \simeq 1.7071$. A more intricate analysis shows that the requirement that $a_{n+1} > 1/2$ is more restrictive than necessary and that even larger values of $\alpha < 2$ would also be feasible.

For example $\alpha = 15/8$ would lead to faster growth of small iterates. Furthermore, if $x_n = 15c/8$, so that $a_n = 15/8$, then $a_{n+1} = 15/64 \simeq 0.23$, $a_{n+2} = 1695/4096 \simeq 0.41$ and the convergence behaves essentially quadratically thereafter. In order that cycles cannot be set up by this process it is necessary that $\frac{15c}{64} > \sqrt{c}$ which in turn implies that $c > (64/15)^2 \simeq 18.2$. Thus the factor $\alpha = 15/8$ *should not* be used with $x_{n-1} \leq 4 < \sqrt{c}$.

There is another good reason for avoiding a factor such as this too soon in RNS, or any integer, arithmetic. The result of the multiplication should be an integer and should be computable using modular operations. With our choice of $Z_1 = 2$, the factor $\alpha = 3/2$ would always result in even integers during this first growth phase and could therefore be repeated. It could also be a precomputed multiplier for modular operations where we know there is no risk of overflowing the dynamic range - which is the present situation. $\alpha = 3/2$ is the largest such constant.

The sequence of initial iterates used until $Z_n > \sqrt{c}$ becomes 2, 6, 54, 4374, ... which with $Z_0 = 2$ is given by $Z_n = 2 \cdot 3^{2^n - 1}$. The maximum number of iterations is reduced to $\lceil \log([1 + \log c]/2\log 3) \rceil$ which is, approximately, equivalent to replacing $\sqrt{c}$ with $c^{(1/3)}$ in the original inequality. With this growth factor for the early iterations our revised algorithm becomes:

Algorithm CRECIP (The accelerated Ceiling version of RECIP)

```
Input:    $Y$
Output:   $\lceil M/Y \rceil$
begin
 $Z_1 \leftarrow 0$
 $Z_2 \leftarrow 2$
 while $Z_1 \neq Z_2$ and $Z_1 - 1 \neq Z_2$ do
   $Z_1 \leftarrow Z_2$
   $Z_2 \leftarrow \lceil Z_1 * (2M - Y * Z_1)/M \rceil$
     $= 2Z_1 - \left\lfloor \frac{YZ_1^2}{M} \right\rfloor$
   if $Z_2 = 2Z_1$ then $Z_2 = \frac{3}{2}Z_1^2$
 return $Z_1$
end.
```

We note that in the second form of the Newton iteration both $2Z_1$ and Z_1^2 have been computed before the test and acceleration step are performed.

5. Numerical experiments. The analysis above suggests that either of the acceleration approaches described in the previous section is likely to represent a significant saving relative to the original modified algorithm of Section 3. Results of tests are included in this section which illustrate clearly that substantial savings in the number of iterations required are indeed achieved by the algorithm CRECIP. The experiments were performed using conventional integer arithmetic with a 32-bit two's complement representation. The number of iterations required for convergence was recorded for each of the algorithms: the modified algorithm of Section 3, the accelerated version using $\alpha = 1$ as described and finally CRECIP with $\alpha = 3/2$.

With either of the accelerated algorithms, the progress will still be slow if the first iterate greater than $\sqrt{c}$ is only slightly greater than this threshold. Whether this is more likely to happen with $\alpha = 1$ or with $\alpha = 3/2$ will vary with c. For any particular value, the fastest convergence will be achieved if an iterate is only slightly smaller than $\sqrt{c}$ so that the next one is close to $2\sqrt{c}$. For this reason there are cases in the results where $\alpha = 1$ appears to be better than $\alpha = 3/2$. However the general result is clearly that the latter choice is to be preferred. As the dynamic range of the integers increases the advantage to be gained from this latter choice would grow.

Specifically, the numbers of iterations were obtained for $c = 2, 3, ..., 10$ and for randomly generated values in each of the intervals $\left[n \cdot 10^k, (n+1) \cdot 10^k\right]$ for $n = 2, 3, ..., 10$ and $k = 1, 2, ..., 8$. Partial results for one particular case form Table 1 and a summary of the performance over several runs of this experiment is presented as Table 2.

The pattern observed even in these partial results is typical of all the results generated. There are cases where the $\alpha = 1$ form of the algorithm performs somewhat better than CRECIP ($\alpha = 3/2$) but the proportional improvement when the latter is best is much greater.

By using random samples such as those described above rather than the full range of integers in the dynamic range, we avoid biasing the results artificially in favor of the CRECIP algorithm due to a preponderance of large values of c. Also by using ”powers of 10” as the basis for the random sample any effect of a pattern in the distribution of the square roots relative to the accelerated iterates powers of 2 or powers of 3 is likely to be minimized.

TABLE 1 Numbers of iterations required for various forms of the integer reciprocation algorithm

c	**Original**	$\alpha = 1$	$\alpha = 3/2$
10	5	5	5
54	9	8	4
94	10	9	7
105	10	9	7
286	12	7	9
4949	17	13	7
8830	18	14	9
53996	21	17	13
90028	21	10	13
723813	25	14	17
29040699	30	19	8
34088349	31	20	10
914885094	36	25	16
1051248160	36	25	16

In Table 2, we present the total numbers of iterates for each of the algorithms for complete runs of the randomized test described above. There were five runs in which all three algorithms were tested and a further three using just the two forms of the accelerated algorithm. All eight of these are taken into account in generating the approximate averages which form the final line of the table.

TABLE 2 Total numbers of iterations for several runs of the randomized test

Original	$\alpha = 1$	**CRECIP**
1648	1097	941
1645	1087	941
1648	1100	942
1649	1090	944
1647	1099	935
Mean 1647	1095	941

In this section we have seen that the use of the ceiling form of the algorithm permits significant savings due to the possibility of generating exact iterate-doubling when current iterates are well-removed from the desired reciprocal. This iterate-doubling also has a simple interpretation in the inference that such iterates are smaller than $\sqrt{c}$. Further analysis of the errors shows that using $\alpha > 1$ is advantageous and because of the requirement for exact integer arithmetic the choice $\alpha = 3/2$ is optimal.

6. Conclusions. In this paper we have presented a modified version of the RNS-integer division algorithm introduced by Hitz and Kaltofen in [6]. This modified version has several advantages over the original unless a very large amount of hardware is available to implement the parallel acceleration technique described in [6].

The basis for the improvements is the use of the ceiling function in place of the floor in the original algorithm. The advantages gained are, firstly, that the additional magnitude comparison required by [6] is avoided since the underlying reciprocation will *always* yield the ceiling of the desired quotient. (Of course, subtraction of unity would deliver the floor if this is required for a specific purpose.) The magnitude comparisons saved involve full MRS conversions and base extensions between the base-RNS and the extension-RNS systems. These are potentially expensive operations.

The second major gain is that the modified algorithm CRECIP lends itself to substantial acceleration of its underlying Newton iteration due to the iterate-doubling inherent in the ceiling form of the integer algorithm. A careful use of the error analysis yields the optimal parameter α for this acceleration scheme. This value is incorporated into the algorithm described here. (It is conceivable that if the value of α is allowed to vary that an even better scheme can be devised.)

The premise for this modification is that in a context such as adaptive beamforming the dynamic range requirement is such that the number of modular arithmetic units required for the hardware-acceleration suggested in [6] is unlikely to be available. The modified algorithm together with suitable pipelining of the linear algebra operations of Gauss elimination (or Gauss-Jordan) could significantly reduce the time-penalty associated with RNS-division. This, in turn, may be sufficient to render the RNS-solution of ABF problems using such elimination algorithms practicable for larger systems than hitherto.

Acknowledgment

This work was supported by grants from the Office of Naval Research through both the Naval Air Warfare Center Aircraft Division, Warminster, PA and the Naval Academy Research Council.

REFERENCES

[1] J.L.Buchanan & P.R.Turner, *Numerical Methods and Analysis*, McGraw-Hill, 1992

[2] W.A.Chren, Jr., *A new residue number system division algorithm*, Computers Math Appl 19 (1990) 13-29.

[3] G.I.Davida & B.Litow, *Fast parallel arithmetic via modular representation*, SIAM J Computing 20 (1991) 756-765.

[4] D.Gamberger, *New approach to integer division in residue number systems*, Proc 10th Symp. on Computer Arithmetic, 1991, 84-91.

[5] M.Griffin, M.Sousa & F.J.Taylor, Efficient scaling in the residue number system, Proc IEEE Conf on Acoustics, Speech and Signal Processing, IEEE, New York, 1989.

[6] M.A.Hitz & E.Kaltofen, *Integer division in residue number systems*, Comp. Sci Tech Report # 93-9, Rensselaer Polytechnic Institute, May 1994

[7] B.J.Kirsch & P.R.Turner, *Modified Gaussian Elimination for Adaptive Beamforming using RNS Arithmetic*, NAWC-AD Tech Report, 94112-50, 1994.

[8] B.J.Kirsch & P.R.Turner, *Adaptive Beamforming using RNS Arithmetic*, Proc ARITH11, IEEE Computer Society, Washington, DC, 1993, pp 36-43.

[9] Mi Lu & J-S Chiang, A novel division algorithm for the residue number system, IEEE Trans Computers 41 (1992) 1026-1032.

[10] N.S.Szabo & R.I.Tanaka, *Residue Arithmetic and its application to Computer Technology*, McGraw-Hill, 1967.

[11] P.R.Turner & B.J.Kirsch, *An Analysis of Gauss Elimination for Adaptive Beamforming*, NAWC-AD Tech Report, 1994.

[12] P.R.Turner & B.J.Kirsch, *Operation complexity for integer or RNS Gaussian elimination*, NAWC-AD Tech Report, 1994.

[13] J.H.Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, 1965.

DISTRIBUTION LIST

No. of Copies

US NAVAL ACADEMY..10
ANNAPOLIS, MD 21402
(ATTN: MATHEMATICS DEPARTMENT, PETER R. TURNER)

AVIONICS DEPARTMENT...12
ENGINEERING DIVISION (CODE 4.5.5.1)
NAVAL AIR WARFARE CENTER
AIRCRAFT DIVISION WARMINSTER
P. O. BOX 5152
WARMINSTER, PA 18974-0591)
(10 FOR CODE 4.5.5.1, BARRY J. KIRSCH)
(2 FOR CODE 7.2.5.5)

DEFENSE TECHNICAL INFORMATION CENTER...............................2
ATTN: DTIC-FDAB
CAMERON STATION BG5
ALEXANDRIA, VA 22304-6145

CENTER FOR NAVAL ANALYSIS...1
4401 FORT AVENUE
P. O. BOX 16268
ALEXANDRIA, VA 22302-0268

OFFICE OF NAVAL RESEARCH..2
800 NORTH QUINCY STREET
ARLINGTON, VA 22217-5660
(2 FOR ONR-313)